Subread User Manual

Version 1.3.1

Wei Shi and Yang Liao

Bioinformatics Division
The Walter and Eliza Hall Institute of Medical Research,

The Department of Computer Science and Software Engineering
The University of Melbourne
Melbourne, Australia

5 April 2013

1 Introduction

Subread is a superfast, sensitive and accurate read aligner, which is designed to align reads generated from the 2nd and 3rd generation sequencers to a reference genome. It supports a variety of sequencing platforms including Illumina GA/HiSeq, ABI SOLiD, Life Science 454, Helicos Heliscope and Ion Torrent. It can align short reads, long reads and reads of variable lengths.

Subread implements a mapping paradigm called "seed-and-vote", which is fundamentally different from the mapping paradigm used by the current generation of read aligners. The "seed-and-vote" paradigm extracts a number of subreads (16 mers) from each read and then use these subreads to vote for the final location of the read, rather than performing a time-consuming extension step. Subread does not perform alignment for the read bases which are covered by the subreads that have made successful votes, because no mismatches are allowed when mapping subreads to the reference genome. Subread performs alignment for those bases which are not covered by the successfully mapped subreads and only align them to the reference bases located at the determined mapping location. This makes Subread an extremely fast aligner. It is 2∼40 times faster than others. Subread achieves high mapping speed without losing sensitivity and accuracy. In fact, it is much more sensitive than other aligners for mapping RNA-seq data because of its highly flexible design. Moreover, Subread has superior accuracy in finding INDEL bases in the reads and it can find up to 16 inserted or deleted bases.

The subjunc program included in the subread package can be used to find exon junction location (for RNA-seq data) or gene fusion locations (for RNA-seq and gDNA-seq data). It returns a BED file which includes the chromosomal locations of exon junctions and/or gene fusions, and a SAM file which records the detailed mapping information for each read (e.g. CIGAR information).

Subread is implemented in C programming language. An R package called "Rsubread', 'which provides wrapper functions for the C function, can be downloaded from: `http://bioconductor.org/packages/2.10/bioc/html/Rsubread.html`.

2 Quick start

A base-space single-end read dataset can be mapped by following the steps below:
Step 1: Install Subread
On a UNIX computer, uncompress the downloaded subread-1.x.x.tar.gz file. Change to the root directory of this package, enter the `src` subdirectory and type `make` command. This will compile the source code and create several executables including `subread-buildindex`, `subread-align` and `subjunc`. These executables will be automatically moved into the `bin` subdirectory located at the root directory. Include the path to these variables to your environment variable "PATH" so that you can access them from anywhere.

Step 2: Build an index
Build a base-space index (default):

```
subread-buildindex -o my_index chr1.fa chr2.fa ...
```

Build a color-space index:

```
subread-buildindex -c -o my_index chr1.fa chr2.fa ...
```

`chr1.fa chr2.fa ...` is a list of names of files which include chromosomal sequences. It typically takes 1 hour to build an index for a human or mouse genome.

Step 3: Perform read alignment
Use the following command to map single-end reads:

```
subread-align -i my_index -r reads.txt -o my_result.sam
```

`reads.txt` is a FASTQ/FASTA format file containing raw read data. `my_result.sam` is a SAM format file and contains mapping results including mapping locations, CIGAR etc.

Use the following command to map paired-end reads

```
subread-align -i my_index -r reads1.txt -R reads2.txt -o my_result_PE.sam
```

Step 4: Discover exon-exon junctions (for RNA-seq)

For single-end data:

```
subjunc -i my_index --singleSAM my_result.sam -o my_junction_result
```

For paired-end data:

```
subjunc -i my_index --pairedSAM my_result_PE.sam -o my_junction_result
```

Output of `subjunc` includes a BED file including discovered exon-exon junctions, and a SAM file including detailed mapping results (eg. CIGAR information for junction reads etc.).

3 Program description

3.1 Build an index for the reference genome

The `subread-buildindex` program builds an base-space or color-space index using the reference sequences. The reference sequences should be in FASTA format (the header line for each chromosomal sequence starts with ">").

This program extracts all the 16 mer sequences from the reference genome at a 2bp interval and then uses them to build a hash table. Keys in the hash table are unique 16 mers and values are their chromosomal locations. By default, the built index is partitioned into blocks each of 3.7 GB in size and at any time only one block is present in the memory. The index built for the human or mouse genome contains two trunks if default

setting is used. Table 1 describes the arguments used by the `subread-buildindex` program.

Table 1: Arguments used by the subread-buildindex program

Arguments	Description
-o < *basename* >	Specify the base name of the index to be created.
-f < *int* >	Specify the threshold for removing uninformative subreads (highly repetitive 16mers). Subreads will be excluded from the index if they occur more than threshold number of times in the reference genome. Default value is 24.
-M < *int* >	Specify the size of requested memory(RAM) in megabytes, 8000MB by default. With default value, the entire built index will be loaded into memory in one go when carrying out read alignment, which enables the fastest mapping speed to be achieved. The actual memory usage is $\sim$ 7600MB for mouse or human genome (other species have a much smaller memory footprint). Requesting less memory will increase running time.
-c	Build a color-space index.
chr1.fa, chr2.fa, ...	Give names of chromosome files.

3.2 Aligning reads to the reference genome

The `subread-align` program extracts a number of subreads from each read and then uses these subreads to vote for the mapping location of the read. The read bases which are not covered by the successfully mapped subreads will then be aligned to the reference bases located at the determined mapping location of the read. Due to the flexible design of the "seed-and-vote" mapping paradigm, `subread-align` program can find mapping locations for the junction reads as well by using part of their sequences for mapping. However, `subread-align` program can not determine either the precise position of the junction breakpoint in the read or the length of introns which the junction reads span. These reads may be passed to the `subjunc` program to get the complete and accurate alignment information (e.g. CIGAR information).

The `subread-align` program allows up to 16 INDEL bases during the mapping for each read, which is a lot more than other aligners. A sophisticated algorithm was designed to determine the location and length of INDEL bases in the read in a more accurate manner. Table 2 describes the arguments used by the `subread-align` program.

Table 2: Arguments used by the subread-align program

Arguments	Description
-i < *index* >	Specify the base name of the index.

-r < *input* >	Give the name of an input file(FASTQ/FASTA format). For paired-end read data, this gives the first read file and the other read file should be provided via the -R option.
-o < *output* >	Give the name of the output file (SAM format).
-n < *int* >	Specify the number of subreads extracted from each read, 10 by default.
-m < *int* >	Specify the consensus threshold, which is the minimal number of consensus subreads required for reporting a hit. The consensus subreads are those subreads which vote for the same location in the reference genome for the read. If pair-end read data are provided, this will be the consensus threshold for the read which receives more votes than the other read from the same pair. 3 by default.
-T < *int* >	Specify the number of threads/CPUs used for mapping. 1 by default.
-I < *int* >	Specify the number of INDEL bases allowed in the mapping. 5 by default.
-P < 3 : 6 >	Specify the format of Phred scores used in the input data, '3' for phred+33 and '6' for phred+64. '3' by default.
-u	Output the uniquely mapped reads only.
-Q	Use mapping quality scores to break ties when more than one best mapping locations are found.
-H	Use Hamming distance to break ties when more than one best mapping locations are found.
-J	Indicate the discovered junction reads in the SAM output using 'S' operation in the CIGAR string. 'Subjunc' program shall then be used to discover exon junction locations and to perform complete alignment for junction reads.
-R < *input* >	Provide the name of the second input file from paired-end read data. The program will then be switched to paired-end read mapping mode.
-p < *int* >	Specify the consensus threshold for the read which receives less votes than the other read from the same pair. This option is only applicable for mapping paired-end read data. The value of this recommended to be smaller than that of '-m' option, so as to rescue the poor quality reads by using the paired-end distance information. 1 by default.
-d < *int* >	Specify the minimum fragment/template/insert length for paired-end read data, 50 by default.
-D < *int* >	Specify the maximum fragment/template/insert length for paired-end read data, 600 by default. Note that the read pairs, which do not meet the distance criteria, may still be reported, provided that one or both reads can be successfully mapped.

-S $< ff : fr : rf >$	Specify the orientation of two reads from the same pair, 'fr' by default (first read is located on forward strand and second read on reverse strand).
-G $< int >$	Specify the penalty for opening a gap when using the Smith-Waterman dynamic programming algorithm to detection insertions and deletions. The Smith-Waterman algorithm is only applied for those reads which are found to contain insertions or deletions. -2 by defaut.
-E $< int >$	Specify the penalty for extending the gap, used by the Smith-Waterman algorithm. 0 by defaut.
-X $< int >$	Specify the penalty for mismatches, used by the Smith-Waterman algorithm. 0 by defaut.
-Y $< int >$	Specify the score for matches, used by the Smith-Waterman algorithm. 2 by defaut.

3.3 Finding exon junction locations

The `subjunc` program maps junction reads by using a large number of subreads extracted from reads to discover the genomic regions spanned by these reads. It then determines the precise spling points in the genome via donor/acceptor site identification and sequence comparison. Mapping location of each base in each read is recorded in the CIGAR field of the generate SAM file.

This program takes as input either a raw read data file (FASTQ/FASTA format) or a SAM file (e.g. SAM output from `subread-align`), and outputs discovered junction locations (chromosomal locations in the reference genome) and complete alignment information of the reads (e.g. CIGAR information). If SAM file is provided as input, this program will use the potential junction reads marked by the aligner (e.g. `subread-align`) to find junction locations and perform alignments for these reads to yield the CIGAR information. If FASTQ/FASTA file is provided, the `subread-align` program will be called to carry out the alignemnt. Table 3 describes the arguments used by the `subjunc` program.

Table 3: Arguments used by the subjunc program

Arguments	Description
-i $< index >$	Specify the base name of the index.
-r $< input >$	Give the name of an input file(FASTQ/FASTA format). Both base-space and color-space read data are supported. For paired-end read data, this gives the first read file and the other read file should be provided via the -R option. 'subread-align' program will be used to align the reads.
-o $< output >$	Give the name of the output file.
-n $< int >$	Give the number of subreads extracted from each read. 14 by default.

–singleSAM $< input >$	Use as input file a SAM file which includes mapping results for single-end reads (e.g. 'subread-align' output)
–pairedSAM $< input >$	Use as input file a SAM file which includes mapping results for paired-end reads (e.g. 'subread-align' output).
-T $< int >$	Specify the number of threads/CPUs used for mapping. 1 by default.
-I $< int >$	Specify the number of INDEL bases allowed in the mapping. 5 by default.
-P $< 3 : 6 >$	Specify the format of Phred scores used in the input data, '3' for phred+33 and '6' for phred+64. '3' by default.
-R $< input >$	Provide the name of the second input file in paired-end data. The program will then be switched to paired-end read mapping mode.
-d $< int >$	Specify the minimum fragment/template/insert length for paired-end read data, 50 by default.
-D $< int >$	Specify the maximum fragment/template/insert length for paired-end read data, 600 by default. Note that the read pairs, which do not meet the distance criteria, may still be reported, provided that one or both reads can be successfully mapped.
-S $< ff : fr : rf >$	Specify the orientation of two reads from the same pair, 'fr' by default (first read is located on forward strand and second read on reverse strand).

4 Citation

Yang Liao, Gordon K Smyth and Wei Shi (2013), "The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote", Nucleic Acids Research, Volume 41, Advance Access, April 4 [Epub ahead of print] (http://nar.oxfordjournals.org/content/early/2013/04/03/nar.gkt214.abstract)

5 Contact

Please email `shi@wehi.edu.au` or `liao@wehi.edu.au` for any inquires.